Concurrent Cell Generation and Mapping for CMOS Logic Circuits

Mineo Kaneko

School of Information Science,
Japan Advanced Institute of Science and Technology
Ishikawa, JAPAN 923–12
Tel: +81–761–51–1276, Fax: +81–761–51–1149
e-mail: mkaneko@jaist.ac.jp

Jialin Tian

Department of Electrical and Electronic Engineering,
Tokyo Institute of Technology
Tokyo, JAPAN 152

Abstract—The conventional technology mapping method is selecting cells from a limited standard library, and the performance of the resultant circuit deeply depends on the characteristics of the library. To realize detailed optimization not limited by an instance of cell library and to reduce the maintenance cost of standard cell libraries, a novel paradigm for technology mapping, in which cell generation and mapping can be executed concurrently, will be considered. This paper shows an outline of a concurrent cell generation and mapping strategy, and proposes a method to map an input Boolean network into CMOS transistor network. The transduction in transistor level is introduced for cell generation and the Dynamic Programming is utilized for cell assignment.

I. Introduction.

The major objectives in the logic circuit design are area minimization, timing optimization and power minimization, and most of the CAD tools employ (1) technology independent optimization of Boolean functions and (2) technology mapping utilizing cell library which has been designed for a particular technology. By the use of such two–stage approach, the entire problem has been moderately partitioned into a set of relatively simple subproblems each of which can be handled with current computer power, and also the CAD tools have acquired a certain level of flexibility with respect to different implementation technologies and the progress of process technology.

Until now, the success of such two–stage approach has been supported by the human efforts to generate and to maintain cell libraries; nevertheless, it is also the fact that the performance of resultant circuits depends partly on the characteristics of the cell library, i.e, the performance is limited by both the limited number of cell functionalities and the limited number of variations along size versus driving–ability tradeoff with respect to each cell functionality.

To keep the certain level of flexibility and to reduce the optimization complexity into a feasible level, the separation of the technology independent optimization and the technology mapping may be reasonable approach, while ideally technology mapping should not be limited by a particular library. To achieve this goal, the user must be able to generate a new gate-cell which realizes a particular cell functionality whenever it is indeed required. The concurrent cell generation and mapping will be an approach to the technology mapping which can escape the critical tradeoff between the performance of a designed circuit and the cost for design / maintenance of the cell library, and hence more detailed optimization can be expected by this approach.

The concurrent cell generation and mapping approach will include following problems; (1) generation of transistor topology for a cell, which will be done concurrently with cell assignment, (2) determination of each transistor size, and (3) generation of physical layout of each cell.

The most important advantage of using pre–designed cell library (conventional case) may be the deeply optimized cell layout and the reliable information about switching delay and fan–out drive–ability with respect to each cell in the library. However, the rapid advance, in recent years, of the technologies for generating cell layout and estimating switching characteristics may possibly overcome this kind of advantage of pre–designed cell library.

As the first step to our concurrent cell generation and mapping approach, the first problem; the cell assignment with cell topology optimization, for CMOS technology will be considered in this paper. The transduction method in transistor level will be proposed for cell generation, and the optimum cell assignment maintained by Dynamic Programming will be investigated. The transduction in transistor level has a potential to escape local optimum and can generate non–serial–parallel structures. Also it has a potential, but not treated in this paper, to handle a sophisticated cost function rather than the number of transistors.

This paper is organized as follows. In section 2, we introduce a cell optimization method called the transduction method, which will be used in cell generation procedure of the technology mapping method. Section 3 describes an approach of technology mapping based on the

ASP-DAC '97
0-89791-851-7$5.00

dynamic programming. While the optimization with respect to various kinds of performance, such as the area, the signal propagation delay and the power consumption will be required to be achieved in the practical situation, the number of transistors will be considered, here in this paper, as the subject to be optimized because of its primitiveness.

II. Transduction in transistor level.

One of the major tasks in the concurrent cell generation and mapping is to generate a cell in transistor level from a given Boolean function. In general, a single output CMOS cell is configured from a two–terminal pMOS transistor network and a two–terminal nMOS transistor network, each of which can be treated as a switching network. *A switching network* is a two–terminal network of switches in which a binary variable is associated with each switch. A switch is usually expressed by an edge in switching networks.

The configuration of switching networks have been treated in several literatures[1]–[3]. However, the resultant network is limited within single pair network or near serial–parallel structure. A straightforward reduction of switching networks is the merging of edges which are connecting to a common node and have a same conductive condition. The method will be easily trapped by a local optimum, even if the permutation of serially connecting edges is regarded with. Sakurai, Lin and Newton[4], Fukui and Newton[5] proposed the method utilizing shared equivalent switch–level network. In any existing methods, the optimization technique is strongly specialized to the objective; the minimization of the number of transistors.

The transduction is a heuristic method which can escape a local minimum state by the use of transformation(add redundancy), and has been introduced for technology independent optimization of multilevel logic circuits. Here we propose the transduction method in switch(transistor) level. The transformation and the reduction, together with the reordering of serially connected sub–networks and the reflection of two-terminal sub–network, will be considered as a set of primitive operations for traversing network space.

A. Initial network from BDD.

An initial switching network can be directly constructed from BDD representation of a target Boolean function(Fig.1). Let $G_{BDD}(V, E)$ be a BDD of a target Boolean function, in which the set of nodes V consists from variable nodes and two constant nodes, v_0 and v_1, as leaf nodes. One variable node is assigned as a root node(v_{root}). Each variable node has two out–going edges, one of which has the label 0 and the other has the label 1.

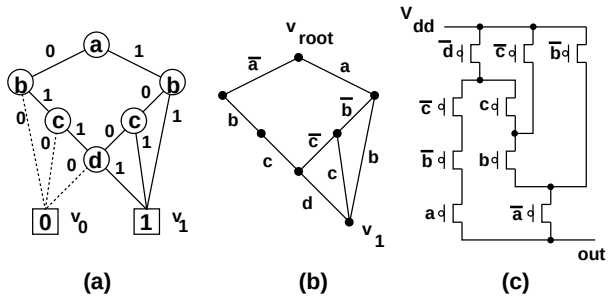

Fig.1. BDD representation, switching network and pMOS network.

[*Construction of switching network for pMOS network*]

1. Remove, from $G_{BDD}(V, E)$, the node v_0 and edges connecting to v_0, and denote the resultant graph as $G_p(V_p, E_p, v_1, v_{root})$.
2. $v_1 \in V_p$ is assigned to a terminal node(V_{dd} (highest voltage) for pMOS network) and $v_{root} \in V_p$ is assigned to the other terminal node(output for pMOS network). The label 0 on an edge (v_i, v_j) with v_i corresponding to an variable x_i is replaced with the conductive condition $\overline{x_i}$(pMOS tr. excited by x_i). Similarly, the label 1 is replaced with the conductive condition x_i(pMOS tr. excited by $\overline{x_i}$).

A switching network for nMOS network is constructed in similar way, but the node v_1 is removed instead of v_0.

It is well known that the size of BDD representation depends on the order of variables, hence the size of the initial switching network depends also on the variable order. The optimization of switching network will mainly rely on the following transduction phase, while, if a smaller size of initial switching network is preferable, the variable order in BDD representation should be optimized.

B. Bases for transformation and reduction.

Basically, when we consider the implementation of fully specified Boolean function, nMOS network and pMOS network can be designed independently. In the following, the transduction for pMOS network will be considered.

Let $\mathcal{F}$ be a completely specified Boolean function to be realized, and let $G_p(V_p, E_p, v_{dd}, v_{out})$ be a pMOS switching network for $\mathcal{F}$. Each edge $e \in E_p$ has a non–inverting or inverting literal as its label which represents the conductive condition for the edge. Two nodes $v_{dd} \in V_p$ and $v_{out} \in V_p$ are specified as terminal nodes.

Definition 1: For Boolean constants 0 and 1, we denote $0 < 1$. For two Boolean functions $\mathcal{F}_1$ and $\mathcal{F}_2$, if either $\mathcal{F}_1 = \mathcal{F}_2$ or $\mathcal{F}_1 < \mathcal{F}_2$ holds for every combination of inputs, then $\mathcal{F}_1$ is said to be contained by $\mathcal{F}_2$ and the relation will be denoted as $\mathcal{F}_1 \leq \mathcal{F}_2$. □

Definition 2: A path which does not include neither v_{dd} nor v_{out} as its internal node is called *conductive path* between two nodes. The product of labels of edges on a

conductive path is called the *conductive condition of the path.* □

Definition 3: The sum of conductive conditions of all conductive paths between two nodes is called *conductive function.* The conductive function between v_{dd} and a node $v \in V_p$ is denoted by $\mathcal{V}(v)$, and the one between a node v and v_{out} is denoted by $\mathcal{O}(v)$. It is clear that $\mathcal{V}(v_{out}) = \mathcal{O}(v_{dd}) = \mathcal{F}$. □

As the set of fundamental operations for modifying switching network, $\mathcal{M}1$: reordering of serially connected two–terminal sub–networks, $\mathcal{M}2$: reflection of two-terminal sub–network, $\mathcal{M}3$: addition of a two–terminal network, $\mathcal{M}4$: merging of two nodes and $\mathcal{M}5$: open–removing of an edge are considered. While $\mathcal{M}1$ and $\mathcal{M}2$ do not change neither the number of edges nor the number of internal nodes, they will enlarge the possibility of further optimization and are commonly used in many literatures. On the other hand, $\mathcal{M}4$ and $\mathcal{M}5$ will contribute to reduce the number of nodes and the number of edges. The modification of switching network achieved by $\mathcal{M}4$ and $\mathcal{M}5$ together with $\mathcal{M}1$ and $\mathcal{M}2$ is unidirectional in the sense that both the number of nodes and the number of edges will be monotonic decreasing, and the traversing will easily deadlock or be trapped at local optimal point. $\mathcal{M}3$ will offer a reverse directional modification which enables us to escape local optimal state and to proceed with a optimization procedure.

The bases for $\mathcal{M}3$ through $\mathcal{M}5$ will be given in the following.

Theorem:($\mathcal{M}3$)
Consider a switching network with $\mathcal{V}(v_{out}) = \mathcal{F}$. When a two–terminal sub–network with x as its conductive function between two terminals is added between two nodes u and v, the conductive function $\mathcal{V}(v_{out})$ is not altered if and only if the following relation holds.

$$x\{\mathcal{V}(u)\mathcal{O}(v) + \mathcal{V}(v)\mathcal{O}(u)\} \leq \mathcal{F} \qquad (1)$$

□

The validity of Theorem:($\mathcal{M}3$) is clear since $x\mathcal{V}(u)\mathcal{O}(v)$ and $x\mathcal{V}(v)\mathcal{O}(u)$ are conductive conditions of conductive paths between v_{dd} and v_{out} which are induced by the addition of the two–terminal network x, and this addition has no other effect on conductive paths in the switching network(Fig.2). Note that the above relation can be rewritten as,

$$x\{\mathcal{V}(u)\mathcal{O}(v) + \mathcal{V}(v)\mathcal{O}(u)\}\overline{\mathcal{F}} \equiv 0 \qquad (2)$$

Hence, the available conductive function x which preserves the cell functionality is given as,

$$x \leq \overline{\mathcal{V}(u)\mathcal{O}(v) + \mathcal{V}(v)\mathcal{O}(u)} + \mathcal{F} \qquad (3)$$

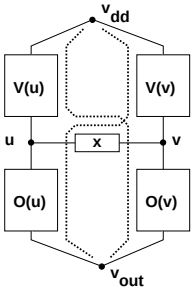

Fig.2. Additional conductive paths induced by the addition of the two–terminal network with its conductive function x.

Corollary 1:($\mathcal{M}3'$)
Single edge can be added between u and v without changing network functionality if and only if $\left\{\overline{\mathcal{V}(u)\mathcal{O}(v) + \mathcal{V}(v)\mathcal{O}(u)} + \mathcal{F}\right\}$ contains a single literal cube. □

Corollary 2:($\mathcal{M}4$)
When two nodes u and v are merged, the conductive function $\mathcal{V}(v_{out})$ is not altered if and only if $\left\{\overline{\mathcal{V}(u)\mathcal{O}(v) + \mathcal{V}(v)\mathcal{O}(u)} + \mathcal{F}\right\}$ is a tautology. □

With respect to $\mathcal{M}5$, the condition for open removing edges can not be formulated with algebraic manner in general, and the conductive function $\mathcal{V}(v_{out})$ will be re–computed to test whether an opening is available or not. (Only if a target switching network is planar, the feasibility of opening an edge will be checked by Corollary2:($\mathcal{M}4$) in the dual switching network.)

It must be important to discuss the completeness of a set of operations for visiting every possible arrangement of switching network, we will bravely skip this issue here in this paper.

C. Cell optimization algorithm.

The minimization of the number of transistors (equivalently the number of edges in the switching network) will be considered as the subject to the optimization.

Basically, the transformation $\mathcal{M}3$ and the reduction $\mathcal{M}4$ and $\mathcal{M}5$ will be performed as one set(transduction phase), while the re–ordering $\mathcal{M}1$ and the reflection $\mathcal{M}2$ of two–terminal sub–networks(re–arrangement phase) will be performed when the above transduction is blocked.

The input of the algorithm is an initial switching network obtained by the direct transformation from BDD representation of a required cell function. First, the initial network is reduced by $\mathcal{M}4$ and $\mathcal{M}5$ procedures until no further reduction can be done, and then the transduction phase follows. The transduction phase will quit when no other redundant edge can be added or when the number of transistors does not decline even after adding a different edge twice(breadth–first search with depth 2).

Fig.3 shows an demonstrative example of the transduction.

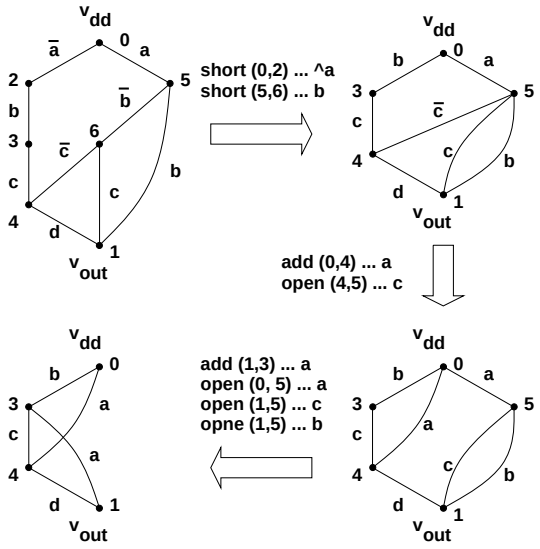

Fig.3. Example of the transduction.

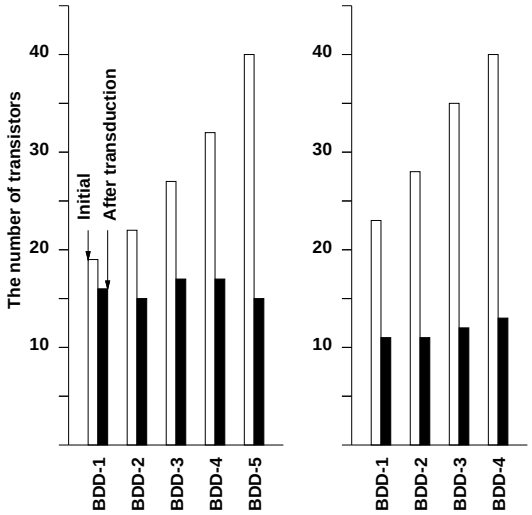

Fig.4. Simulation results begin with various BDD representations.

In the transformation $\mathcal{M}3$ within the transduction phase, the choice of an edge to be added will be performed with the following heuristic rules for reducing the search space.

1. Single transformation will be limited to single edge addition (use Corollary 1). In other ward, the transformation which increases the number of edges will be adopted, while the one which increases the number of nodes will not be employed.
2. If the function $\mathcal{F}$ is monotonic increasing (decreasing) with respect to a variable y, then only the edge $y(\overline{y})$ will be regarded for addition.
3. The edge x such that $x\{\mathcal{V}(u)\mathcal{O}(v) + \mathcal{V}(v)\mathcal{O}(u)\} \equiv 0$ will be excluded from the addition, since no new conducting path will be created by this transformation, and no further reduction will be expected.

The computational complexity will be briefly studied. The evaluation will be performed in terms of the number of BDD operations. Let m and b be the number of nodes and the number of edges in the initial switching network. It takes $O(mb)$ for computing $\mathcal{V}$ and $\mathcal{O}$ of all nodes in the network. $O(b \cdot mb)$ for visiting every edge and testing $\mathcal{M}5$. $O(m^2)$ for visiting every node pair for $\mathcal{M}4$. Since $\mathcal{M}4$ and $\mathcal{M}5$ will be repeated for every possible "single edge addition" and there are m^2 node pairs, it takes, in worst case, $O(m^2) \times (O(m^2) + O(mb^2)) = O(m^4 + m^3b^2)$ to find valid "single edge addition" by which and by the following reduction the size of the network is reduced. The body of the most outer loop will be repeated at most b times, hence totally it takes $O(b) \times O(m^4 + m^3b^2) = O(m^4b + m^3b^3)$.

Fig.4 shows simulation results for two different Boolean functions. The purpose of this simulation is to check the effect of the choice of initial network to the final network. Each of two graph indicates the number of transistors in the initial network and resultant network starting at various BDD representations of the same Boolean function(i.e., BDD representations with various ordering of variables).

We can conclude that the number of edges in the resultant network has less dependency with the size of initial network.

III. Cell Mapping.

The input to the cell mapping is a Boolean network with multiple inputs and multiple outputs. We assume that the Boolean network has been optimized by a technology independent multi–level optimization procedure.

A successful and efficient solution for the technology mapping was suggested by Kurt Keutzer, and it is implemented in DAGON[6] and MIS[7]. The basic idea of both is converting the technology mapping problem into *Directed Acyclic Graph*(DAG) covering problem and solving DAG covering by a sequence of tree covering which can be performed optimally using a dynamic programming algorithm. Basically we will follow this method.

We will use a DAG to represent a Boolean network, in which nodes are labeled with the name of a primitive gate such as AND, OR, NAND, NOR, etc. The directed edge (v_i, v_j) represents a connection between th output of v_i and input of v_j. A primary input (denote as PI or a leaf) has no in–coming edge. A primary output(PO or a root) has no out–going edge.

In our cell mapping considered here, any cell library will not be used, but CMOS cells will be generated concurrently with "tree covering".

A. Tree covering based mapping.

The application of the tree covering to technology mapping proceeds as follows. At first, all nodes with 3 or more than 3 inputs in a given Boolean network are converted

into 2–input nodes(*technology decomposition*), and then the network is partitioned into a forest of trees by breaking at each multiple–fanout point. The resultant trees are optimally covered by one tree at a time. Finding the optimum covering of a subject tree is done with generating two kinds of optimum cells, i.e., one is nominal–phase cells and the other is complementary–phase cells, and selecting the optimum match from among the candidates using a dynamic programming algorithm.

With respect to the tree covering procedure, important features of our method(concurrent cell generation and mapping) to conventional technology mapping using cell library are;

1. A canonical form for representing both a subject DAG and cell functionality of each cell in the cell library, such as NAND–network representation or 2–input–NAND–network representation, is not necessary. Our technology decomposition will be applied to the input Boolean network itself for only enlarging the search space.
2. Since we do not have any cell library, "matching" will be defined in function level not in pattern level. That is, for each node v_i in a subject DAG, we will extract various sub–trees rooted at the node v_i and an associated cover F_i for each sub–tree will become the input to the cell generation procedure. Then the output (generated cell) of the cell generation procedure is treated as the "match" for the corresponding sub–tree rooted at node v_i.
3. It is well known that the functionality of CMOS cell is limited to a monotonic decreasing function with respect to its inputs, and the best insertion of inverters is unknown beforehand. Then, we will take the strategy that, for an extracted cover F_i of each sub–tree, two optimized cells, one realizes F_i and the other realizes $\overline{F}_i$, will be generated. In compensation for generation overhead, we can extract sub–trees without the limitation of the polarity of inputs.

Given a circuit after partitioning, a dynamic programming is used for the mapping to achieve the least number of transistors.

Now we consider the search front to reach at a node v_i of a tree T. We first generate a set of candidate matches for the node in such way;

[*Generation of a set of candidate matches*]
(1) Extract a set of candidate sub–trees rooted at v_i,
(2) An optimum CMOS cell will be generated for the associated cover F_i for each sub–tree,
(3) An optimum CMOS cell will be generated for the complement of the associated cover $\overline{F}_i$ for each sub–tree,
(4) The set of all generated CMOS cells forms the set of candidate matches for the node v_i.

A match generated from F_i is called a "nominal match", while one from $\overline{F}_i$ is called a "complementary match".

Since the CMOS cell with the functionality F_i is implemented from the switching network for F_i as its pMOS network and the one for $\overline{F}_i$ as its nMOS network, the generation of the optimized CMOS cell for $\overline{F}_i$ seems trivial once after the optimized CMOS cell for F_i is generated. It must be true when only the number of transistor is our concern, however when the other criteria such as the switching delay are to join to cost function, the switching network optimized for pMOS network and the one of the same functionality optimized for nMOS network may be different.

For every match at v_i, the cost of an optimal cover containing that match will be defined as the sum of the cost of the corresponding CMOS cell and the sum of the costs of the optimal covers for the nodes which are inputs to the match. In the following, the cost of the optimal cover for a tree rooted at v_i is considered to be associated with the node v_i.

It is well known that a CMOS cell can realize a monotonic decreasing function with respect to its inputs, and appropriate insertion of inverters(equivalently, polarity inversion of internal variables) on a given Boolean network is required for completing a cover (CMOS cell mapping). To take care of such inverter insertion dynamically in the dynamic programming algorithm, two kinds of costs are maintained, one of which is called the "nominal–phase cost" and the other is called the "complementary–phase cost".

Definition 4:
The *nominal–phase cost* at a node v_i(denoted as $COST_p(v_i)$) is the cost of optimal cover for the tree rooted at the node v_i, by which cover the nominal functionality associated with v_i is realized. On the other hand, the *complementary–phase cost* at a node v_i (denoted as $COST_n(v_i)$) is the cost of optimal cover for the tree rooted at the node v_i, by which cover the complement of the functionality associated with v_i is realized. □

Now we let $cost_p(v_i)$ be the cost of an optimal cover for the tree rooted at v_i among the covers containing a nominal match of v_i, and let $cost_n(v_i)$ be the one among the covers containing a complementary match of v_i. Also we let $cost_{inv}$ be the cost of a inverter cell. Then,

$$COST_p(v_i) = \text{MIN}\,[cost_p(v_i), cost_n(v_i) + cost_{inv}] \quad (4)$$
$$COST_n(v_i) = \text{MIN}\,[cost_n(v_i), cost_p(v_i) + cost_{inv}] \quad (5)$$

Note that, in evaluating the cost of an optimal cover containing a match of a node, the cost of each optimal cover for the node(say v_j) which is an input to the match is either $COST_p(v_j)$ or $COST_n(v_j)$, which depends on the request from the generated CMOS cell corresponding to the match. That is, when the variable driving that CMOS cell and the one nominally associated to the node v_j have the same polarity, $COST_p(v_j)$ is used, while $COST_n(v_j)$ is used when they have the different polarities. If the

CMOS cell requires both polarities of an input variable, MAX $(COST_p(v_j), COST_n(v_j))$ is used.

B. Implementation and results.

While the optimization with respect to various kinds of performance, such as the area, the signal propagation delay and the power consumption will be required to be achieved in the practical situation, the number of transistors will be considered, here in this paper, as the subject to be optimized because of its primitiveness.

At the practical implementation of the design procedure, the following are regarded.

1. In the case of conventional technology mapping utilizing cell library, the search space is limited by the number of patterns in the pattern set of the cell library, while our method has no such limitation and we can always find a corresponding CMOS cell for every extracted sub–tree. Hence, in order to reduce the search space(and also it may contribute to avoid CMOS cells having too long conductive paths, but this issue had better to be discussed in the delay driven optimization problem), two restrictions on choosing sub–trees are imposed, one of which is the number of fanin, and the other is the height of sub–tree.
2. In order to reduce the execution time, a cost table of simple cells such as NAND, NOR, is prepared in advance.

Table 1 shows the result of design examples†. For each input data(single output Boolean network), CMOS networks are synthesized with various parameter set with respect to the maximum height and the maximum number of fanin for sub–trees in making match.

Table 1. Design examples

Data(Gates)	H.L.	F.L.	#Tr.s	CPU(s)
data1(20)	2	4	64	0.530
data1(20)	3	6	58	3.490
data1(20)	4	10	52	6.130
data1(20)	5	15	48	12.62
data2(50)	2	4	136	1.130
data2(50)	3	6	122	9.330
data2(50)	4	10	108	18.47
data2(50)	5	15	98	38.51
data3(97)	2	4	260	3.020
data3(97)	3	6	230	20.60
data3(97)	4	10	202	37.87
data3(97)	5	15	184	77.20

H.L.: Height limitation, F.L.: Fan–in limitation
#Tr.s: Number of transistors
CPU(s): SUN Sparc 2 workstation

†Complete design system including automated *decomposition and partitioning* is under developing.

While the cell assignment concurrent with cell (transistor topology) generation has been proposed in this paper, it should be followed by the determination of each transistor size and the generation of physical layout of each cell. For our new paradigm to compete with the library based approach, these tasks should be completely automated with sufficient performance in area/delay optimization and with reasonable design time.

IV. Conclusions.

We have proposed a framework of the concurrent cell generation and mapping for CMOS logic circuits. The transduction in transistor level is also proposed for generating optimal CMOS cells. While only the number of transistors is considered, in this paper, as the subject to be optimized, the proposed method possesses the enough flexibility to deal with other various performances to be optimized.

The concurrent cell generation and mapping dealing with not only the number of transistors but also the signal propagation delay and power consumption and the connection with a transistor size optimizer and a cell layout generator remain as major problems. The utilization of internal don't cares in the Boolean network is also an interesting problem, since, now, the optimization of pMOS network and the one of nMOS network will not be achieved independently but interrelatedly each other.

Acknowledgement

The authors would like to thank Mr. Tetsuya Asano(currently, with Hitachi Co.) and Prof. Mahoki Onoda, Takusyoku University, for their helpful discussions. This research is a part of CAD21 Project at Tokyo Institute of Technology, and the authors would like to thank all members of CAD21 for their supports.

References

[1] M.Y.Wu, W.Shu, S.P.Chan, "A unified theory for MOS circuit design switching network logic", *Int. J. Electron.*, vol.58, no.1, pp.1–33, 1985.

[2] M.Y.Wu, I.N.Hajj, "Switching network logic approach to sequential MOS circuit design", IEEE Trans. Computer Aided Design, nol.8, no.7, pp.782–794, 1989.

[3] J.Zhu, M.Abd-El-Barr, "On the optimization of MOS circuits ", IEEE Trans. Circuits and Systems–I, vol.40, no.6, 1993.

[4] T.Sakurai, B.Lin, A.R.Newton, "Multiple–output shared transistor logic(MOSTL) family synthesized using binary decision diagram", UCB/ERL Report, 1990.

[5] M.Fukui, A.R.Newton, "Optimum module generation for semi–custom design", Proc. Asia–Pacific Conf. Circuits and Systems, pp.184–189, 1992.

[6] K.Keutzer, "DAGON:technology binding and local optimization by DAG matching", Proc. 24–th Design Automation Conf., pp.341–347, 1987.

[7] E.Detjens, G.Gannot, R.Rudell, A.Sangiovanni–Vincentelli, A.Wang, "Technology mapping in MIS", Proc. Int. Conf. CAD, pp.116–119,1987.